In the claims:

1. (currently amended) A computer system for transferring ~~data~~ messages between a receiving central processing unit (CPU) and a transmitting CPU by using only write operations therebetween for the purpose of avoiding a direct read operation between said receiving CPU and said transmitting CPU, said system comprising:

a) at least one receiving central processing unit (CPU) comprising at least a read head register, a first queue length register, and a first total read register;

b) at least one transmitting CPU comprising at least a write head register, a second total read register, a total write register, and a second queue length register;

c) a local memory for said receiving CPU;

d) a local memory for said transmitting CPU;

e) means for connecting between said receiving CPU and ~~second~~ said transmitting CPU where such means ~~transfer~~ transfers write operations faster than read operations; ~~and~~

f) a circular queue defined between designated addresses in said local memory of said receiving CPU, wherein said read head register contains a pointer to the location of the next read from said circular queue and said write head register contains a pointer to the location of the next write into said circular queue;

g) means for periodically updating said second total read register with the content of said first total read register; and

h) means for adding and updating at least a message separator between messages, wherein a read operation of said receiving CPU is achieved when said transmitting CPU performs:

1) a write operation providing a separator to said local memory of said receiving CPU at a location pointed to by said write head register; and
2) a write operation of at least one message to said local memory of said receiving CPU at a location pointed to by said write head register.

2. (deleted)

3. (deleted)

4. (currently amended) The system of claim 1 wherein said means for connecting between said ~~CPUs~~ said receiving CPU and said transmitting CPU is a PCI bus.

5. (deleted)

6. (deleted)

7. (deleted)

8. (deleted)

9. (deleted)

10. (currently amended) The system of claim ~~9~~ 1 ~~where~~ wherein said ~~pointer for the next~~ read head register ~~from said queue~~ and said ~~pointer for the next~~ write head register ~~into said queue~~ are set to point to the same address upon initialization.

11. (currently amended) The system of claim ~~10~~ 1 ~~where~~ wherein a maximum length is ~~specified for said~~ imposed on a message

to be written into said circular queue.

12. (currently amended) The system of claim 11 ~~where~~ wherein a tail is added at the end of said circular queue, said tail being equal in length to said maximum length ~~for said~~ imposed on said message.

13. (currently amended) The system of claim ~~12~~ 1 ~~where~~ wherein a ~~message~~ header separator is used to indicate the end of said message.

14. (currently amended) The system of claim 13 ~~where~~ wherein said ~~message~~ header separator contains the length of ~~said~~ the immediately following message.

15. (currently amended) The system of claim 13 ~~where~~ wherein said ~~message~~ header separator contains a predefined header "magic" number.

16. (original) The system of claim 15 ~~where~~ wherein, if said ~~message~~ header separator contains an erroneous header "magic" number, an error message is generated.

17. (currently amended) The system of claim 13 ~~where~~ wherein ~~said message~~ a separator of the last message in said queue is a stopper ~~designator~~ separator, and is different from said ~~message~~ header separator placed between subsequent messages.

18. (currently amended) The system of claim 17 ~~where~~ wherein said stopper ~~designator~~ separator further contains a predefined stopper "magic" number.

19. (currently amended) The system of claim 18 ~~where~~ wherein, if a stopper ~~designator~~ separator contains an erroneous stopper "magic" number, an error message is generated.

20. (withdrawn) A method for writing a data message into a receiving queue between memory segments separated by a data bus comprising the steps of:

a) checking that the length of said message is not greater than the length of said queue, and generating an error message if said message length is greater than said queue length;

b) checking that length of said message is not greater than a maximum message length, and generating an error message if said message length is greater than said maximum message length;

c) repeatedly checking if there is sufficient memory available in said queue to contain said message until such time that sufficient memory is available;

d) writing said message into said queue;

e) writing a stopper designator immediately after the end of said message;

f) replacing the stopper designator at the end of the immediately preceding message with a message separator; and

g) updating the content of a total writes register by adding the number of bytes written into said queue.

21. (withdrawn) The method of claim 20 where said availability of memory in said queue is checked by a transmitting CPU by comparing said message length with the difference between said queue length and the difference between the total data written and total data read.

22. (withdrawn) The method of claim 20 further comprising aligning said message to a predefined alignment scheme by adding alignment padding bytes at the end of said message.

23. (withdrawn) The method of claim 22 where said alignment is on a four byte granularity.

24. (withdrawn) The method of claim 22 further comprising assigning the new address of the write head by calculating the sum of the old write head plus said message length plus any applicable padding, plus the lengths of said beginning message separator and said stopper designator.

25. (withdrawn) The method of claim 22 where, in the event that said data message was first written into the tail of said queue, calculating said new address of write head further comprises deducting said the queue length.

26. (withdrawn) A method for reading a data message from a transmitting CPU into a queue with a tail of a receiving CPU, comprising:

a) checking that said message to be read is in said queue or otherwise wait for said message to enter said queue;

b) checking that the "magic" number is valid, or otherwise generate an error message; and

c) reading said message without alignment padding bytes.

27. (withdrawn) The method of claim 26, further comprising calculating the number of said alignment padding bytes added to said message.

28. (withdrawn) The method of claim 27, further comprising calculating the new address of the read head as the sum of the old read head plus the length of the message, plus

alignment padding, plus the lengths of the stoppers, in the event that said data read is not from said tail of said queue.

29. (withdrawn)The method of claim 28 wherein said calculating further comprises deducting said length of said the queue, when said data was read from said tail of said queue,.

30. (withdrawn)The method of claim 27 wherein said number of alignment padding bytes is added to the contents of the total read register.

31. (withdrawn) The method of claim 30 wherein the content of the total read register is written into the corresponding register of the memory of said receiving CPU.

32. (withdrawn) The system of claim 1, wherein writing a data message into said circular queue comprises the steps of:

a) checking that the length of said data message is not greater than the length of said queue, and generating an error message if said data message length is greater than said circular queue length;

b) checking that length of said data message is not greater than a maximum message length, and generating an error message if said message length is greater than said maximum message length;

c) repeatedly checking if there is sufficient memory available in said queue to contain said message until such time that sufficient memory is available;

d) writing said data message into said queue;

e) writing a stopper designator immediately after the end of said message;

f) replacing the stopper designator at the end of the immediately preceding message with a message separator; and

g) updating the content of a total writes register by adding the number of bytes written into said circular queue.

33. (withdrawn) The system of claim 12, wherein reading a data message from a transmitting CPU into said queue having said tail, comprises the steps of:

a) checking that said data message to be read is in said queue or otherwise wait for said message to enter said queue;

b) checking that the "magic" number is valid, or otherwise generate an error message; and

c) reading said message without alignment padding bytes.